UNIVERSITY OF AVEIRO

DEPARTMENT OF ELECTRONIC, TELECOMMUNICATIONS AND INFORMATION

INTEGRATED MASTER IN COMPUTER AND TELEMATICS ENGINEERING

Software Engineering

M3 SPEC- Full Specification and Requirements

2015/2016

Group: g04

João Almeida, 50640 Ricardo Silva, 50427 Hugo Silva, 49900

Index

Project description

The goal of this project is to develop a standalone system for reading Portuguese Citizen SmartCards, logging the events (card inserted/removed) and info (card number/name of owner/…) both locally and in an external server (through a message broker), allowing the visualization of the card data and logging history in a locally provided web page and establishing an authentication method which connects each card to a user defined password.
This system is to be implemented as part of the “DETI - Events and rooms manager” project and will provide information to the “Entry Control” module relative to the card events registered.

Project Requirements

Hardware Requirements:

- SmartCard Reader (model used during development: http://www.bit4id.com/en/index.php?option=com_content&view=article&id=86&Itemid=503&lang=en)

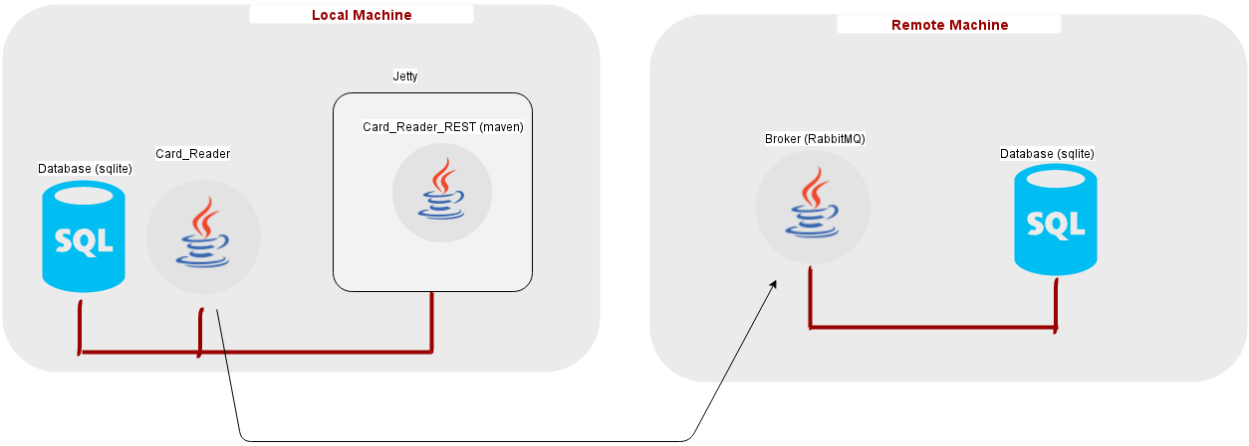
Software Requirements:

- Linux OS (not really necessary but the system is not yet optimized for Windows as well)
- Java (Oracle) JDK1.8 (http://java.com/en/download/manual.jsp)
- Portuguese citizen card middleware and certificates (https://www.cartaodecidadao.pt/index.php_option=com_content&task=view&id=102&Itemid=44&lang=pt.html)
- RabbitMQ service installed and started/enabled (https://www.rabbitmq.com/download.html)
- Sqlite3 (http://mislav.net/rails/install-sqlite3/)
- Jetty Servlet Engine (http://download.eclipse.org/jetty/)

(All java library dependencies are already included along with the Jar and War packages provided.)

Project Modules

This project is composed by 4 different modules:

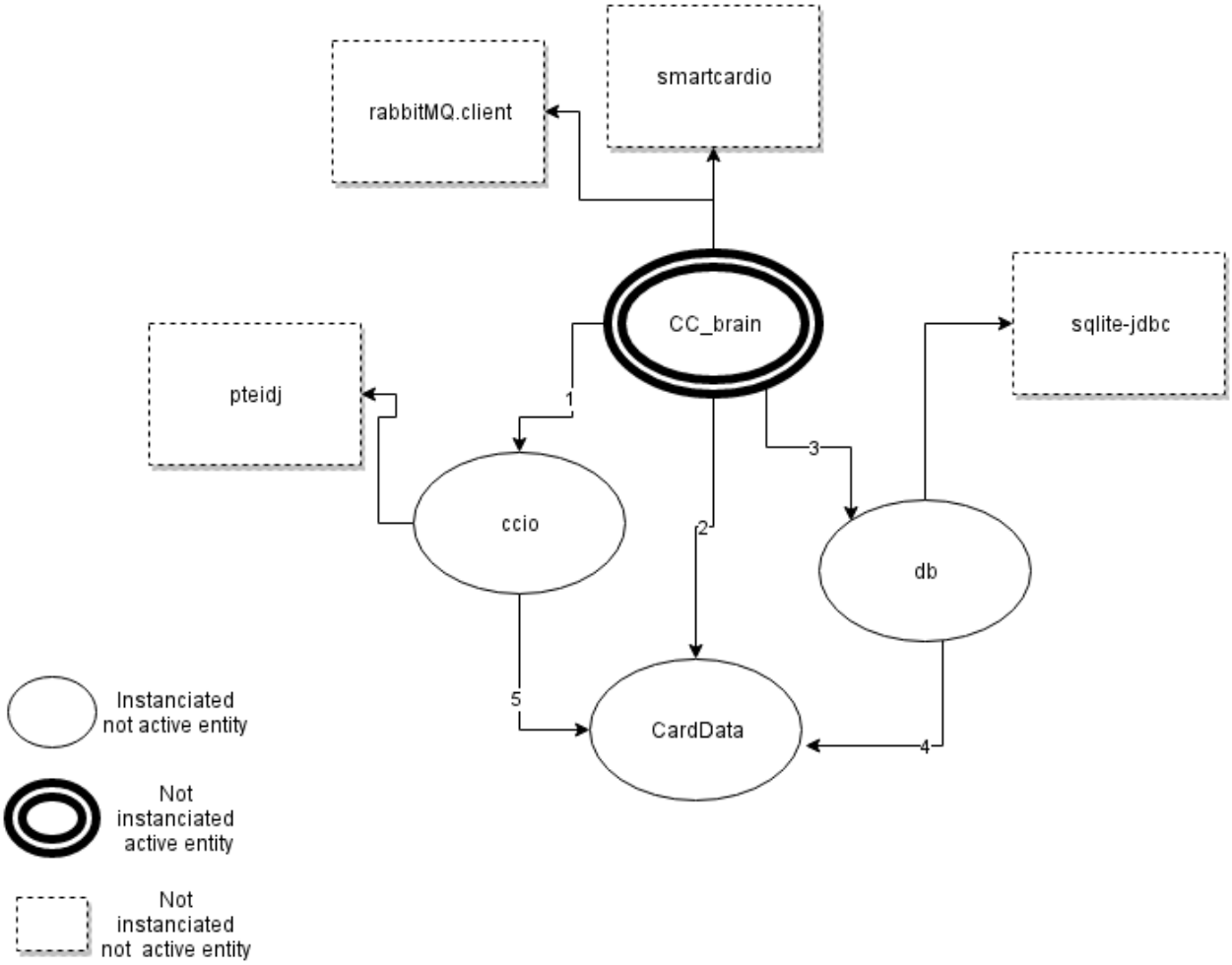


- Database

A sqlite Database which stores the list of people that used the Card Reader and their interactions:

- Card Reader

Java application which listens for events relating to the smartCard reader (inserted/removed), logs them in the local database and sends them to the broker, in the form of a rabbitMQ message (with a JSON String as content)



Methods:

1 - instantiate,
RunAnalisys(String photoPath) -> gets the information of the card inserted in the reader and returns a CardData object with it (null if exceptions occur).

2 - instantiate,
getNumBI() -> returns the value of numBI variable in the CardData Object.

getJson(String roomCode, String interaction) -> returns String with Json object containing information about the card and the interaction event (room code, inserted/removed, timestamp).

3 - instantiate,
connect(String PathToDB) -> establishes connection to database
connection_close() -> closes connection to database
dump_interaction(CardData card, String roomCode, String interaction) -> inserts event in database and, if the card was inserted for the first time, adds person to database.
update_curent_card(String person_id) -> updates id of the card currently inserted in the reader to the database table current_card, this information is used by the web page application.

4 - instantiate,
get…() -> return value of private variable

5 - instantiate

- Local Web Page

Maven application deployed by the Jetty Servlet Engine the provides a local interface (in the internet browser) for the owner of the citizen card currently inserted in the reader to consult his/her information as well as the local logging history associated with the card.

Jetty Server specifications:
Port: 3389
User Page Path: “/CC_Reader_REST/UserPage_servlet”

- Imaginary Broker

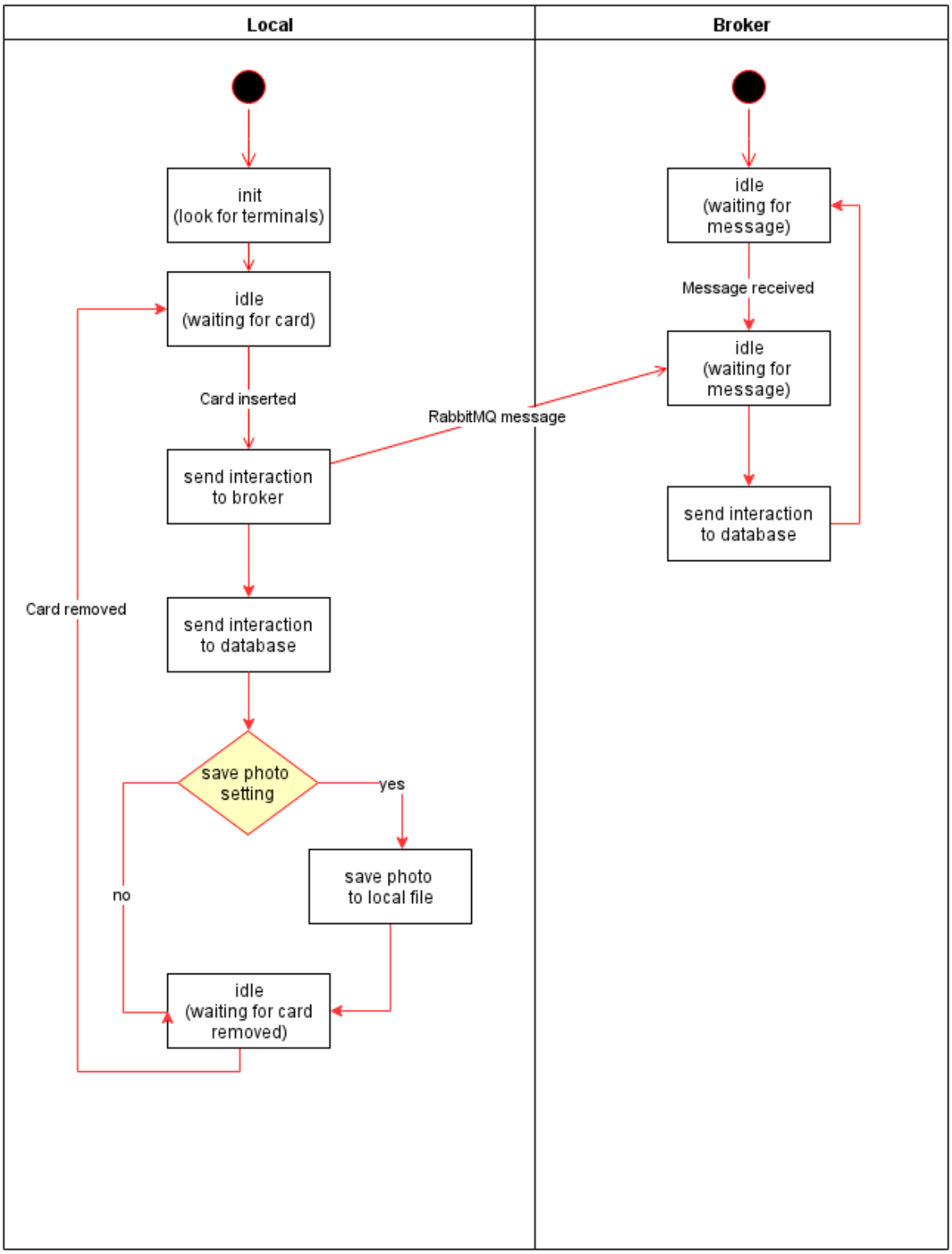
This imaginary broker was used to simulate the interaction the system will have with the “Entry Control” module in a posterior development of the “DETI - Events and rooms manager” project .
So far, the broker is connected to a replica database identical to the one used by the main system and logs the users and interactions with the information received from the messages sent by the Card_reader app.
The message body consists of a Json String with the user info and the interaction occurred:

Example of the json file:

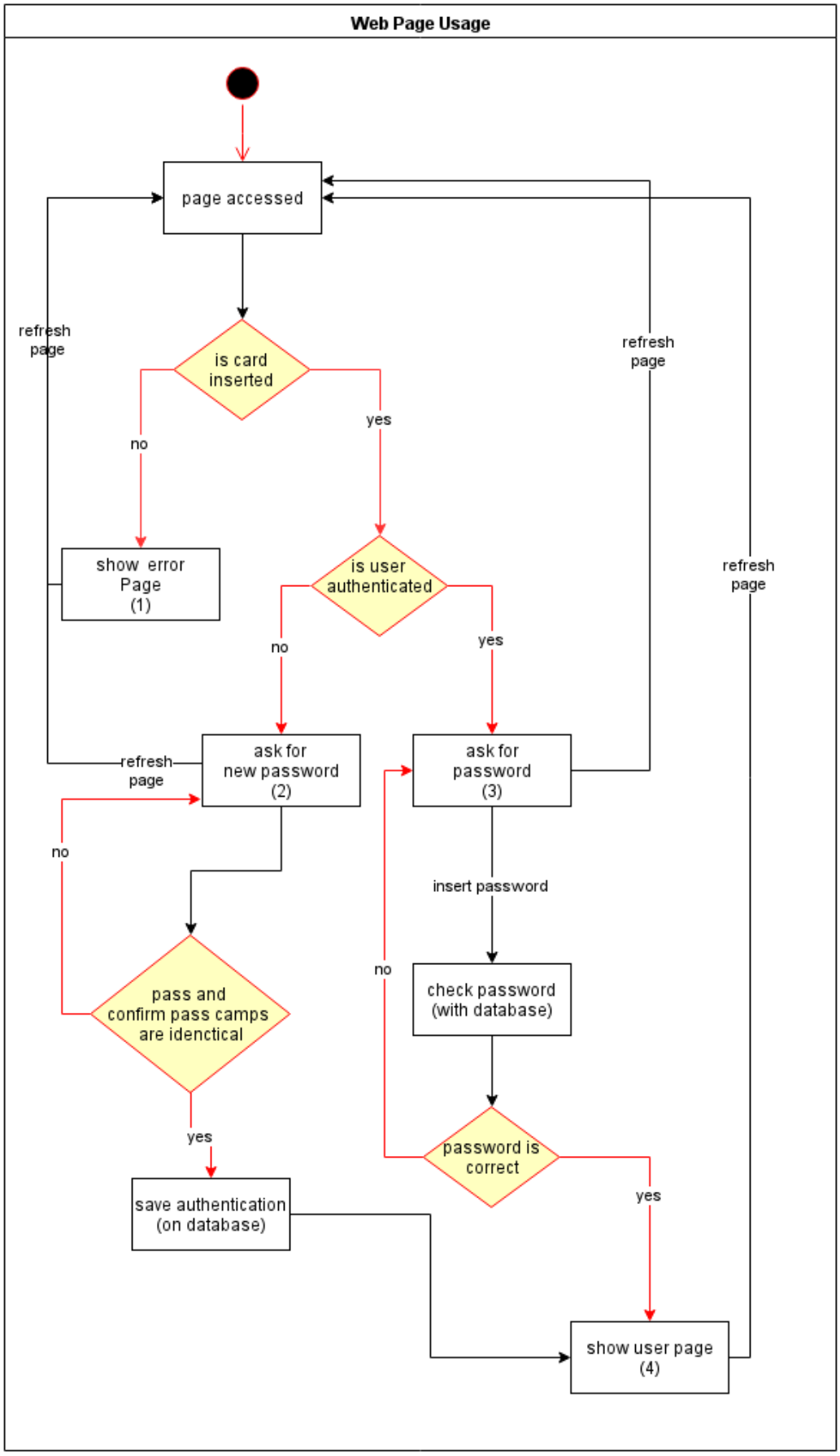

```
{
    "interaction_info":
    {
        "interaction":"inserted",
        "roomCode":"4.2.11",
        "time":1459202118914
    },
    "person_info":
    {
        "country":"PRT",
        "firstname":"TIM",
        "notes":"",
        "documentType":"Cartão de Cidadão",
        "cardVersion":"004.004.20",
        "numBI":"123456789",
        "lastnameFather":"TEST",
        "lastnameMother":"TEST",
        "numSNS":"111111111",
        "firstnameMother":"MOTHER",
        "locale":"Test Street - AVEIRO",
        "deliveryDate":"12 12 2014",
        "height":"1,70",
        "numSS":"11111111111",
        "sex":"M",
        "cardNumberPAN":"0000011111111111",
        "firstnameFather":"FATHER",
        "birthDate":"11 01 1990",
        "mrz3":"TTSRRAG<SSDFG<<TIMTIMT<TTTTT<<",
        "mrz2":"111111111111113PRT<<<<<<<<<<<4",
        "lastname":"TEST",
        "mrz1":"U<PRT111111111<1111<<<<<<<<<<<",
        "nationality":"PRT",
        "numNIF":"111111111",
        "cardNumber":"11111111 0 ZYB",
        "deliveryEntity":"República Portuguesa"
    }
}
```

Activity Diagrams

- Card Reader Cycle:



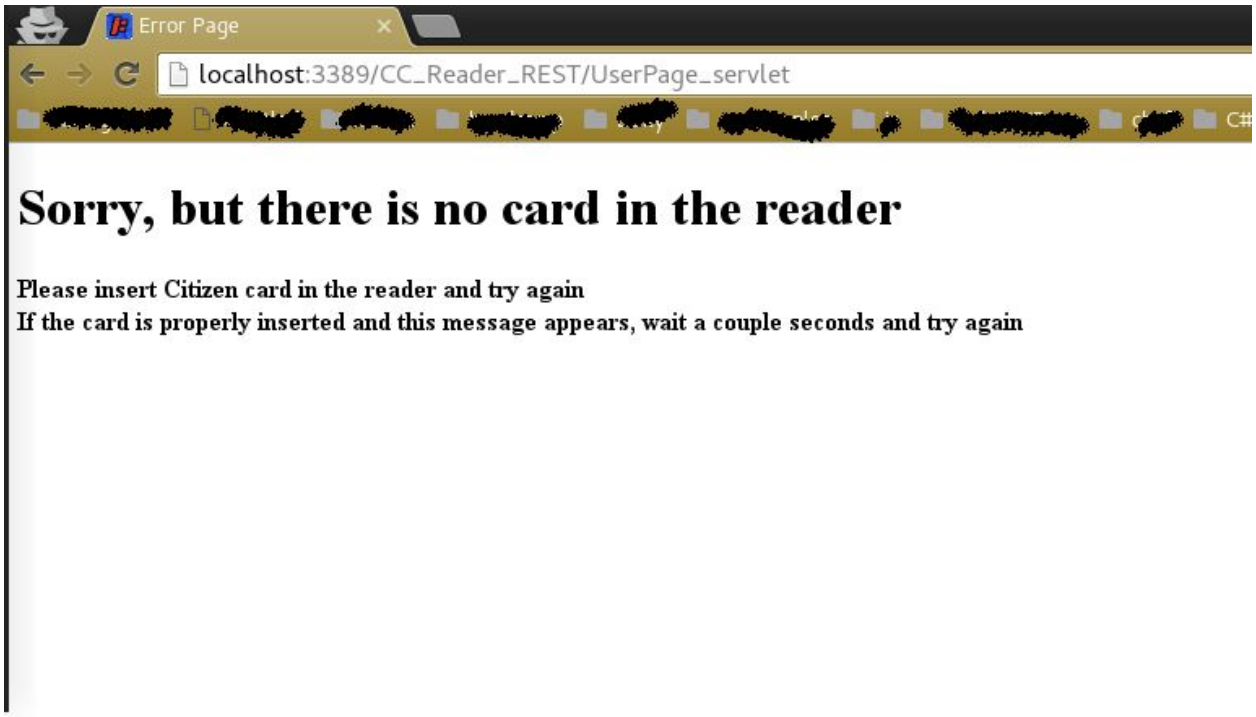
- Web Page usage:



Pages:

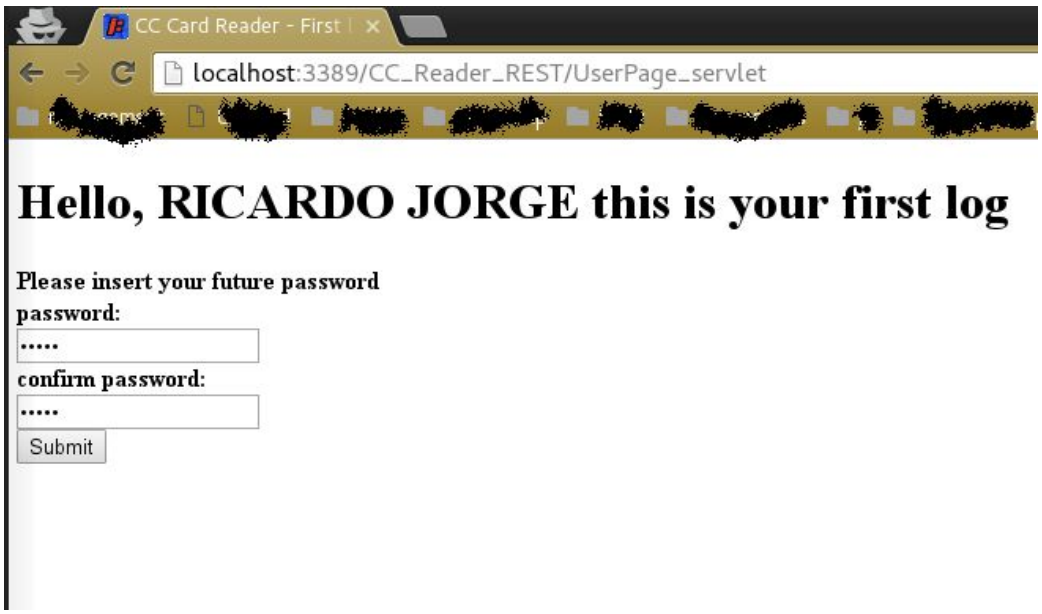
(1) Error Page

This page is displayed when there is no card in the reader or the card hasn't been processed yet.



(2) First Login Page

If the card is inserted but the user still hasn't authenticated his/her account, the server prompts the user to define the password which will be used in the future to access the personal data.



(3) Nomal Login

If the card is inserted and the user already has a password associated to his/her account,the system requires that password in order to move on to the User Data page.

(4) User Page

After the credentials have been given, the system displays the user's personal data as well as his/her logging data for card related events in the local machine.

Features

- User data and event logging (both locally and remotely through rabbitMQ messaging)
- User account authentication through user defined password
- Local Web Server deployable through Jetty
- System building and deploying executable in one command using bash script (in Linux) (script names: build_and_deploy.sh / deploy.sh).

Use Cases

The use cases defined are only relative to SmartCard owner interactions.

ID: UC-01
Title: register personal account on database
Description: add user data to database and log first interaction (card inserted)
Preconditions: system running, citizen card not expired
Postconditions: user becomes registered in database and can authenticate account
Main Steps:

1 - User inserts card in reader

2 - waits a few seconds (while application processes card information)

3 - user is registered. (optional: if user page is accessed in the browser and “first login” information is displayed, this serves as confirmation of successful registering.)

Possible errors:

2 - card is removed unexpectedly or is corrupted -> register action is canceled.

Frequency of use: frequent at start but decreasing with time.
Status: working.

ID: UC-02
Title: authenticate account on database
Description: establish password associated with user account on database
Preconditions: system running, citizen card inserted in reader, user registered
Postconditions: user authenticates account and becomes able to enter User Page
Main Steps:

1 - User goes to User Page URL in browser (localhost:3389/CC_Reader_REST/UserPage_servlet)

2 - “first login” page is displayed

3 - user fills “password” and “confirm password” camps and submits.

4 - user’s account is now authenticated and User Page is displayed

Possible errors:

3 - “password” and “confirm password” camps don’t match, the system asks for the credentials to be inserted again.

Frequency of use: frequent at start but decreasing with time.
Status: working.

ID: UC-03
Title: Check personal information
Description: consult personal card data and logging history.
Preconditions: system running, citizen card inserted in reader, user registered and authenticated
Postconditions: none.
Main Steps:

1 - User goes to User Page URL in browser (localhost:3389/CC_Reader_REST/UserPage_servlet)

2 - “login” page is displayed

3 - user fills “password” camp and submits.

4 - User Page is displayed

Possible errors:

3 - “password” camp doesn’t match the password stored in database, the system asks for the credentials to be inserted again.

Frequency of use: frequent.
Status: working.

Handicaps

In this section we will list the known bugs affecting the system and some of the betterments to be developed in posterior iterations.

Bugs:

- Wrong type of card inserted (gets caught in try catch and doesn't break program, but system stops being able to read citizen cards properly after this occurs)

UnderDevelopment Modules:

- Authentication remains local (because imaginary broker is only receiving messages and not replying, a user can have different passwords for different machines)
  Possible solutions
  - Responsive broker that, when receiving event checks for changes in the user's password and returns it.
  - Local broker (using a local broker in each machine that listens for "password changed" messages from remote server)

  The final solution must be discussed with other groups.
  Priority: High

- Security issues (database is accessible by anyone using the computer in which the system is running)
  Possible Solutions:
  - Ciphered database (using hardcoded password

  Priority: High

- Ugly interface (the web page interface can be better)
  Priority: Low

- Password changing (so far, the password defined by the user in the first login is final, but it should be redefinable at any time)

Priority: Medium